

Joint Deep Learning for Car Detection

Seyedshams Feyzabadi
Electrical Engineering and Computer Science
University of California, Merced
sfeyzabadi@ucmerced.edu

Abstract

Traditional object recognition approaches apply feature extraction, part deformation handling, occlusion handling and classification sequentially while they are independent from each other. Ouyang and Wang proposed a model for jointly learning of all of the mentioned processes using one deep neural network. We utilized, and manipulated their toolbox in order to apply it in car detection scenarios where it had not been tested. Creating a single deep architecture from these components, improves the interaction between them and can enhance the performance of the whole system. We believe that the approach can be used as a general purpose object detection toolbox. We tested the algorithm on UIUC car dataset, and achieved an outstanding result. The accuracy of our method was 97 % while the previously reported results showed an accuracy of up to 91 %. We strongly believe that having an experiment on a larger dataset can show the advantage of using deep models over shallow ones.

1. Introduction

Object recognition plays an important role in computer vision, and has been extensively discussed in the literature [20, 5, 26, 21]. The main difficulty in creating a robust object detection approach comes from the wide range of variations in images of objects belonging to the same object class. Similar objects from the same class might have different sizes, different shapes because of the view points etc. Having a part-based model helps to overcome these issues and create an abstract model of the object by using its parts. Similar challenges of different textures, lighting, background and etc. increase the difficulties of detecting an object in a picture.

The traditional approach of overcoming these challenges follows the following steps:

1. Feature extraction: features are extracted from an image to find out the most discriminative parts of it. The

most famous approaches are SIFT [22] or Haar [27]

2. Deformation handling: extracting multiple parts of an object will be very helpful when there are variety of shapes for the same object while they consist of similar part. For instance, cars have different shapes, but they all consist of body, tires, trunk and hood. Agarwal performed a good research on part-based object detection [1, 2].
3. Occlusion handling: Occluded parts have to be detected in order to avoid using them in object detection itself. Including them in the detection phase will reduce the accuracy of the program. Some usages of occlusion detection algorithms can be found here [28, 8].
4. Classification: After finishing all the procedures, a classifier decides whether the chosen window contains an instance of that object or no. The most known approaches are SVM [6] and random forests [7].

Most of the traditional approaches follow these steps sequentially. Each of them is created and trained first, and the output of each step is fed as the input to the next step. However, the interactions and feedbacks of each step to the other one has not been studied well.

Ouyang et al [24] proposed the first model for joint learning of all of these steps in one single deep convolutional network and reported good results in pedestrian detection scenarios. Their model is based on learning all of the steps at the same time, and use the output of higher levels in training the lower levels. Figure 1 shows their approach for joint deep learning in pedestrian detection. Their method is capable of using the training data in creating better low level filters. This vividly explains why the joint learning can be beneficial comparing to independent learning.

In this paper we focus on Ouyang's work [24] for pedestrian detection and mostly base our work on their efforts. The main contribution of this paper is to generalize their framework and apply it on other applications. The author strongly believes that this approach has the potential to become a generic framework for object recognition. The

biggest advantage of this approach is its flexibilities where it makes the approach very easy to change application. We can imagine that the same method can be applied to other objects too. We also believe that the deformation layer of this methodology makes it more useful for objects that are composed of many parts. As an example a bicycle can be a good representative object. Another interesting application can be in animal detection where their body parts are not rigid and may have different articulations.

2. Related Work

Several approaches to object detection were proposed in the past that use some form of learning. In most such approaches, images are represented by using some features, and a learning method is used to identify regions in the feature space that correspond to the object class. There are a large variety in types of features used and the learning methods applied [11].

Recently deep models showed great performance in machine learning and recognition problems [18, 14, 17, 19, 25, 9] comparing to shallow approaches. There are some robotics usages of deep and shallow models too [13, 10, 4]. They proved and showed that these approaches are capable of solving complex tasks. Their complicated structure makes them more compatible for complex problems with a lot of details. But, it arises the problem of unknown regions. There is not any proof why they are so efficient [18], how they can be improved or what is a structured way of creating the best model.

Most of the traditional approaches use features such as SIFT or HOG [6] to extract the overall shape of an object, and apply different learning approaches to train the system. All of these approaches have one common point that the features are manually generated, and only used for the training purposes. While some recent investigations show that learning features from the training data is very helpful and improves the accuracy of the program [3].

Detecting cars is an important aspect in traffic control scenarios. It will be very useful to distinguish cars from other objects. Multiple approaches have focused on this problem [12, 1, 2]. Since different types of cars have different shapes, but are composed from similar parts, it will be very beneficial to create a part-based model for it. Then the same solver can be applied to different classes of cars such as sedans, SUVs, trucks and etc. Comparing to other objects, cars have very huge variety of shapes and sizes. Thus it will be too hard, if not impossible, to have a general model for all of them. But in this work, we only focus on sedan cars.

We propose a model for car detection that benefits from deep learning approaches and is capable of detecting different classes of cars. The approach uses the training data to improve its low-level features. Even though we applied this

approach to side view pictures of cars, it is extendable to other views of the cars too. We very closely followed the work by Ouyang [24] and used the same methodology to detect different object class.

3. Our Approach

As discussed earlier, the main difference between this method with traditional recognition method is the jointly learning of the mentioned processes. Its architecture is shown in Figure 2. In general, the following steps are performed in this program:

- The first convolutional layer generates filtered data maps. This layer convolves the 3-channel input image data with $9 \times 9 \times 3$ filters and outputs 64 maps.
- Average pooling is applied to Features maps with the size of 64 filtered data maps using 4×4 boxcar filters with a 4×4 subsampling step.
- The second convolutional layer generates part detection maps. This layer convolves the feature maps with 8 part filters of different sizes and outputs 8 part detection maps.
- Part scores are obtained from the 8 part detection maps using a deformation handling layer. This layer outputs 8 part scores.
- The visibility reasoning of 8 parts is used for estimating the label y ; that is, whether a given window encloses a car or not.

3.1. Input Data

The size of an input image is assumed to be 84×28 and it is fed to the program with the following three channels:

- The first contains Y data in YUV format. Its size is 84×28 .
- The three-channel 42×14 images in the YUV color space are appended into the second channel of size 84×28 with zero padding.
- Four 42×14 edge maps are appended into the third channel of size 84×28 . Three edge maps are obtained from the three-channel images in the YUV color space. The magnitudes of horizontal and vertical edges are computed using the Sobel edge detector [15]. The fourth edge map is chosen as the maximum magnitudes from the first three edge maps.

Ouyang et al, found it useful to feed the program in three channels as explained above, and we follow their strategy. They claimed that

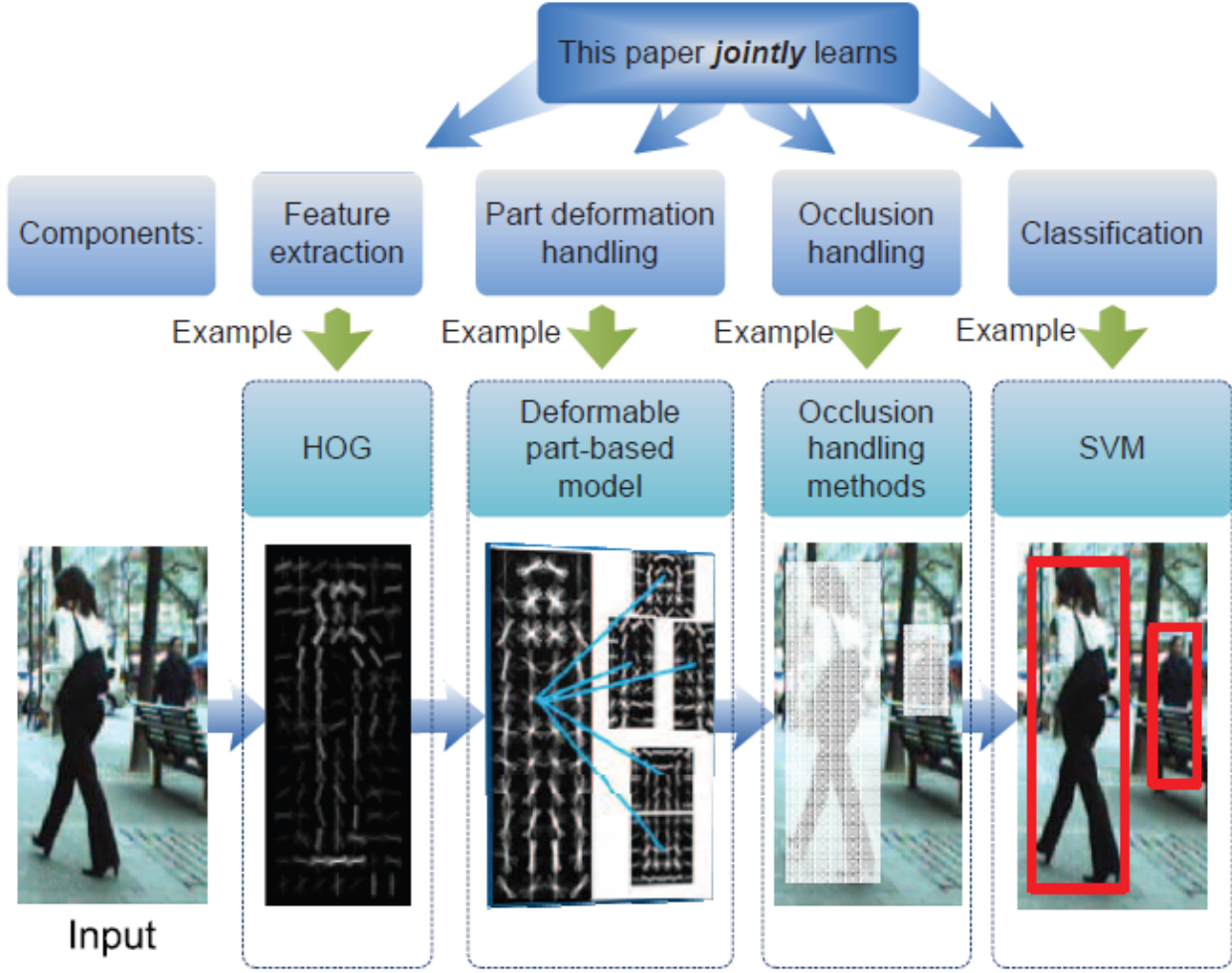


Figure 1. Joint deep learning for pedestrian detection, taken from [24]

In this way, information about pixel values at different resolutions and information of primitive edges are utilized as the input of the first convolutional layer to extract features. The first convolutional layer and its following average pooling layer use the standard CNN settings. We empirically find that it is better to arrange the images and edge maps into three concatenated channels instead of eight separate channels. In order to deal with illumination change, the data in each channel is preprocessed to be zero mean and unit variance [24].

3.2. Part Maps

Most of the vision approaches have filters with equal sizes, but cars might have different sizes or been seen from different distances.. Therefore, we decided to have variable size filter as shown in Figure 3. The filters are defined on

three levels and the size of the largest one is 5 x 15. The smaller ones have portion of the whole filter. The side view image of a car can be taken from both sides, therefore, it is needed to have filters for both views. The left and right side of Figure 3 deal with this issue.

3.3. Deformation Layer

In this subsection we would like to explain what happens in the second convolutional layer. This layer receives the P part detection maps as input and part scores $s = \{s_1, \dots, s_P\}$, $P = 8$, are its output. The convolutional layer treats the detection maps individually and produces the p th part score s_p from the p th part detection map, denoted by M_p . A 2D summation map, B_p , is calculated by summing up the part detection map M_p and the deformation maps as follows:

$$B_p = M_p + \sum c_{n,p} D_{n,p}$$

$D_{n,p}$ denotes the n th deformation map for the p th part,

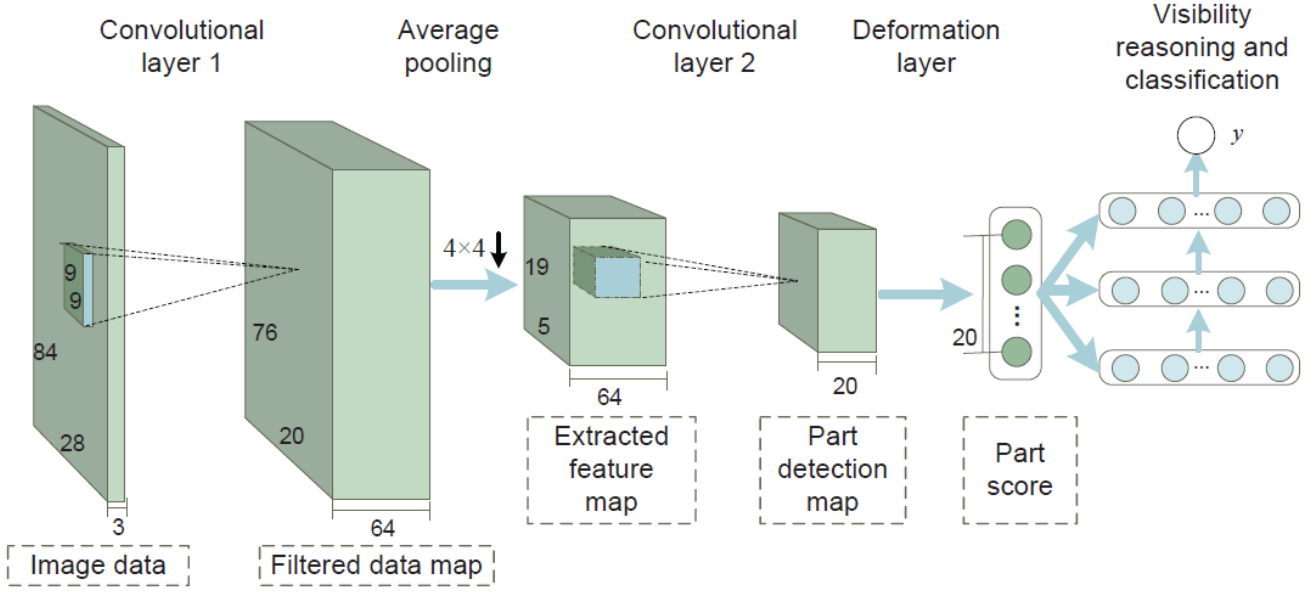


Figure 2. Program architecture, taken from [24]

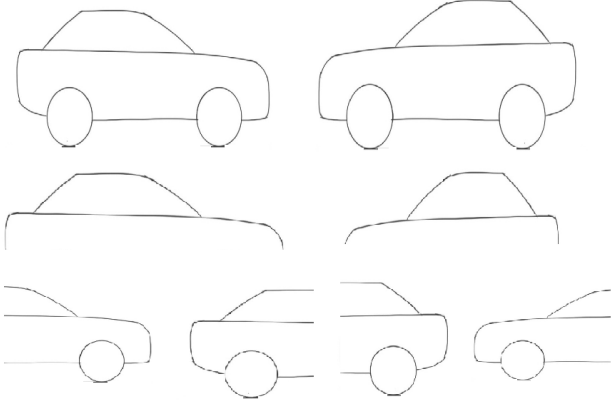


Figure 3. Part filters for the side view pictures of cars.

$c_{n,p}$ denotes the weight for $D_{n,p}$, and N denotes the number of deformation maps. s_p is globally max-pooled from B_p

$$s_p = \max b_p^{(x,y)}$$

where $b_p^{(x,y)}$ denotes the (x, y) th element of B_p . The detected part location can be inferred from the summed map as follows:

$$(x, y)_p = \operatorname{argmax}_p b_p^{(x,y)}$$

At the training stage, only the value at location $(x, y)_p$ of B_p is used for learning the deformation parameters.

The $c_{n,p}$ and $D_{n,p}$ are the key for designing different deformation models. Both $c_{n,p}$ and $D_{n,p}$ can be considered as the parameters to be learned. Three examples are given below:

1. Suppose $N = 1$, $c_{1,p} = 1$ and the deformation map $D_{1,p}$ is to be learned. In this case, the discrete locations of the p th part are treated as bins and the deformation cost for each bin is learned. $d_{1,p}^{(x,y)}$, which denotes the (x, y) th element of $D_{1,p}$, corresponds to the deformation cost of the p th part at location (x, y) .
2. $D_{1,p}$ can also be predefined. Suppose $N = 1$ and $c_{n,p} = 1$. If $d_{1,p}^{(x,y)}$ is the same for any (x, y) , then there is no deformation cost. If $d_{1,p}^{(x,y)} = -\infty$ for $(x, y) \in X$, $d_{1,p}^{(x,y)} = 0$ for $(x, y) \in X$, then the parts are only allowed to move freely in the location set X . Max-pooling is a special case of this example by setting X to be a local region. The disadvantage of max-pooling is that the hand-tuned local region does not adapt to different deformation properties of different parts.
3. The deformation layer can represent the widely used quadratic constraint of deformation. Below, we skip the subscript p . The quadratic constraint of deformation can be represented as follows:

$$b^{(x,y)} = m^{(x,y)} + c_1 \left(x - a_x + \frac{c_3}{2c_1} \right)^2 + c_2 \left(y - a_y + \frac{c_4}{2c_2} \right)^2 \quad (1)$$

where $m^{(x,y)}$ is the (x, y) th element of the part detection map M , (a_x, a_y) is the predefined anchor location of the p th part. They are adjusted by $c_3/2c_1$ and $c_4/2c_2$, which are automatically learned. c_1 and c_2 decide the deformation cost. There is no deformation cost if $c_1 = c_2 = 0$. Parts are not allowed to move if $c_1 = c_2 = \infty$. (a_x, a_y) and $(\frac{c_3}{2c_1}, \frac{c_4}{2c_2})$ jointly decide the center of the part. The quadratic constraint can be represented as follows:

$$B = M + c_1 D_1 + c_2 D_2 + c_3 D_3 + c_4 D_4 + c_5 \cdot 1, \quad (2)$$

$$b^{(x,y)} = m^{(x,y)} + c_1 d_1^{(x,y)} + c_2 d_2^{(x,y)} + c_3 d_3^{(x,y)} + c_4 d_4^{(x,y)} + c_5,$$

$$d_1^{(x,y)} = (x - a_x)^2,$$

$$d_2^{(x,y)} = (y - a_y)^2,$$

$$d_3^{(x,y)} = x - a_x,$$

$$d_4^{(x,y)} = y - a_y,$$

$$c_5 = \frac{c_3^2}{4c_1} + \frac{c_4^2}{4c_2}$$

where 1 is a matrix with all elements being one, $d_n^{(x,y)}$ is the (x, y) th element of D_n . In this case, c_1, c_2, c_3 and c_4 are parameters to be learned and D_n are predefined. c_5 is the same in all locations and need not be learned.

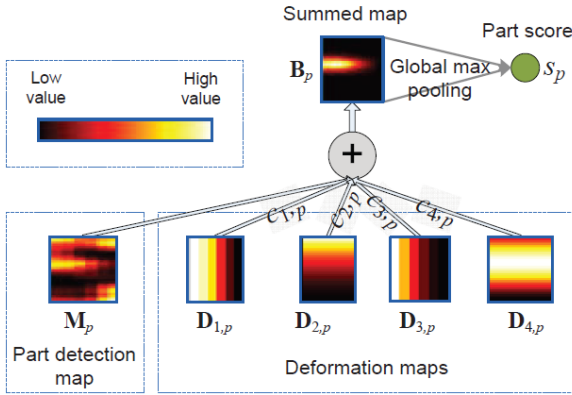


Figure 4. The deformation layer. Part detection map and deformation maps are summed up with weights $c_{n,p}$ for $n = 1, 2, 3, 4$ to obtain the summed map B_p . Global max pooling is then performed on the summed map to obtain the score s_p for the p th part. Taken from [23].

Figure 4 is an example of learning these parameters. In

this case parameters c_1, c_2, c_3 and c_4 are to be learned and D_n is predefined. It is notable that a large part of this subsection follows the work of Ouyang.

3.4. Classification

The results of previous section is a set of scores s_1, s_2, \dots, s_n . Then visibility reasoning is used for classification. More details of it can be found in [23].

In this paper, features, deformable models, and visibility relationships are jointly learned. Backpropagation through s is used as prediction in order to learn the parameters in the two convolutional layers and the deformation layer in Figure 2.

In order to train this deep architecture, we used a multi-stage training strategy. We start with a 1-layer CNN using supervised training. Since Gabor filters [16] are similar to the human visual system, they are used for initializing the first CNN. We add one more layer at each stage, the layers trained in the previous stage are used for initialization and then all the layers at the current stage are jointly optimized with back propagation.

4. Experimental Results

The described architecture is tested on UIUC database¹. The database contains 1050 images (550 cars and 500 non-cars). The test set consists of 170 images with 200 cars in them. The file format of the images are PGM. There are some evaluation codes in the dataset which we did not use for this experiment, but they can be utilized for evaluating different object recognition algorithms. A snapshot of the data set is shown in Figure 5.

Image sizes are 40×100 and they are in gray scale. In order to be in line with Ouyang's work, we rotated and scaled the images to 84×28 . Therefore, we can use the same set of filter as them.

We used log average miss rate in order to make comparisons with other methods. It is computed by averaging the miss rate at nine FPPI rates that are evenly spaced in the log-space in the range from 0.01 to 1.

In order to improve the results we applied augmentation. This is done by creating manual rotations on the images starting from -10 degrees up to +10 with intervals of 1 degree.

As a comparison method we picked the method developed by Agarwal et al [1]. They used the same data set with similar test images. Their method is robust and accurate and a sample results of their method is shown in Figure 6. Their average miss rate is 9 %.

The results of our experiments without augmentation is shown in Figure 7. However, the results is not disappointing. It is not too bad either. The miss rate of this approach

¹<http://cogcomp.cs.illinois.edu/Data/Car/>



Figure 5. UIUC dataset. The upper part shows the positive images, and the lower part is the negative samples.

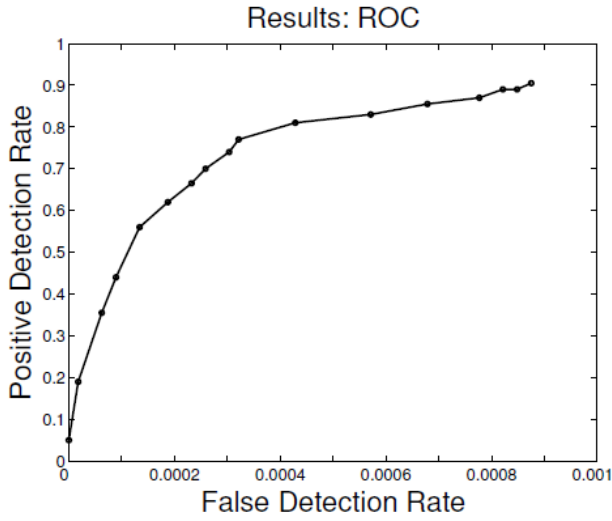


Figure 6. The results of Agarwal's work, published in [1].

is 23 %.

Augmentation is needed to improve the dataset and makes it more stable against camera angles. After we applied the augmentation, we achieved the results in Figure 8.

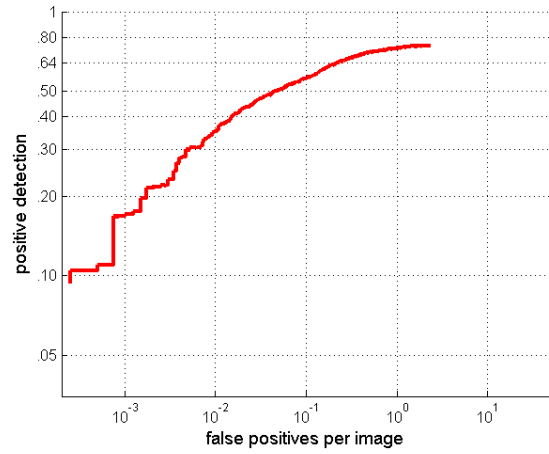


Figure 7. Jointly learning without Augmentation.

This method decreased the average miss rate to 3 % which shows a great improvement comparing to the reported 9 % from Agarwal's work.

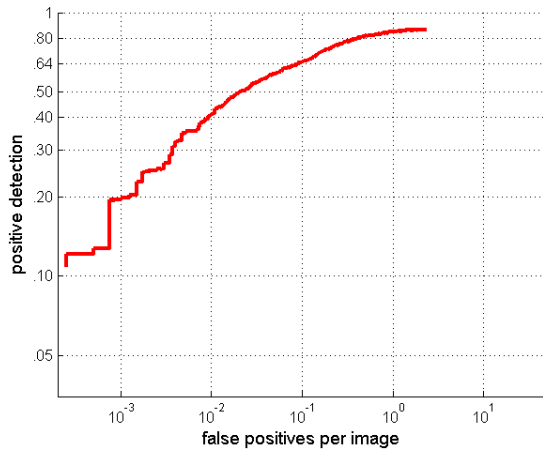


Figure 8. The results of our algorithm after applying augmentation.

5. Conclusions

In this report we showed our work about using joint deep learning architecture for car detection applications. The main idea of joint deep learning is to include feature extraction, part deformation handling, occlusion handling and classification in one single deep neural network where two layers of convolutional layers exist. The first layer extracts feature maps using low-level features that are tuned during the training phase. And the second convolutional layer is responsible for deformation handling to extract a score based on different visible parts of an object.

We applied the algorithm on UIUC car dataset, and managed to gain good results of up to 97 % of success rate, after applying augmentation. Comparing our results with older results of Agarwal, our method outperforms the Agarwal's result by 6%. We also showed the importance of augmentation in the final performance of an algorithm. Since deep learning approaches require large datasets, we increased the size of our database by applying augmentation.

References

- [1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1475–1490, 2004.
- [2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Computer Vision/ECCV 2002*, pages 113–127. Springer, 2002.
- [3] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *Computer Vision–ECCV 2010*, pages 127–142. Springer, 2010.
- [4] P. Baumann, S. Feyzabadi, and C. Jucovski. Putting pixels in place: A storage layout language for scientific data. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 194–201. IEEE, 2010.
- [5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.
- [6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [7] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision–ECCV 2012*, pages 645–659. Springer, 2012.
- [8] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *Computer vision and pattern recognition (CVPR), 2010 IEEE Conference on*, pages 990–997. IEEE, 2010.
- [9] A. Ess, B. Leibe, and L. Van Gool. Depth and appearance for mobile scene analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [10] S. Feyzabadi, S. Straube, M. Folgheraiter, E. A. Kirchner, S. K. Kim, and J. C. Albiez. Human force discrimination during active arm motion for force feedback design. *Haptics, IEEE Transactions on*, 6(3):309–319, 2013.
- [11] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.
- [12] C. Goerick, D. Noll, and M. Werner. Artificial neural networks in real-time car detection and tracking applications. *Pattern Recognition Letters*, 17(4):335–343, 1996.
- [13] S. H. Hamraz and S. S. Feyzabadi. General-purpose learning machine using k-nearest neighbors algorithm. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 529–536. Springer, 2006.
- [14] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [15] B. Jähne, H. Scharf, and S. Körkel. Principles of filter design. *Handbook of computer vision and applications*, 2:125–151, 1999.
- [16] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. In *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pages 14–19. IEEE, 1990.
- [17] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [19] Q. V. Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

- [20] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [21] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(4):349–361, 2001.
- [22] P. C. Ng and S. Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.
- [23] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3258–3265. IEEE, 2012.
- [24] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2056–2063. IEEE, 2013.
- [25] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Un-supervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [26] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.
- [27] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 734–741. IEEE, 2003.
- [28] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.